Underwater Computer Vision of the ZEABUS AUV

Supakit Kriangkhajorn
Department of Computer Engineering
Kasetsart University
Bangkok, THAILAND
supakit.kr@ku.th

Akrapong Patchararungruang
Department of Computer Engineering
Kasetsart University
Bangkok, THAILAND
akrapong.p@ku.ac.th

Somchai Numprasertchai
Department of Computer Engineering
Kasetsart University
Bangkok, THAILAND
somchai.n@ku.ac.th

***Abstract*—this paper presents the implementation of underwater computer vision of the ZEABUS Autonomous Underwater Vehicle (AUV) at Kasetsart University. The purpose of the implementation is to augment previously used algorithm for the 2018 International RoboSub Competition and Singapore AUV 2018 Challenge. The result shows that new algorithm has higher precision level and can achieve better object detection.**

Keywords—computer vision, image processing, underwater computer vision, autonomous underwater vehicle

I. INTRODUCTION

Underwater computer vision is one of challenging field. The main confrontations of underwater perception are due to high device cost, complex setup, and distortion in signals and light propagation introduced by the water medium [1]. However, it is an important part of an Autonomous Underwater Vehicle (AUV) because it is a major mission the AUV is made for. For example, the system to identify pipeline structure placed on the sea bottom, detect possible obstacles (e.g. trestles), and detect and recognize some landmark objects, i.e., anodes [2]. Moreover, the computer vision is used to evaluate the AUV position along the mission plan. In ZEABUS team, the underwater computer vision is separated into 2 sections consisting of hardware and software. In hardware section, we use two sets of Imaging Development Systems (IDS) UI-3260CP-C-HQ camera. One is mounted with Kowa LM6HC 6 mm lens and the other is equipped Tamron M111FM08 8 mm lens. The cameras are mounted on the front and bottom of AUV. In software section, we use Robot Operating System (ROS) [8] for connecting to our camera and Python with OpenCV for implementing the algorithms and verifying our ideas.

In this paper, section II shows the camera configuration. The image enhancement method is illustrated in section III. Section IV depicts the method to convert a grayscale image to the binary formatted. Section V and VI describe the color detection method and the shape detection algorithm respectively. The experimental results are shown in section VII. Then, the conclusion of the paper.

II. CAMERA CONFIGURATION

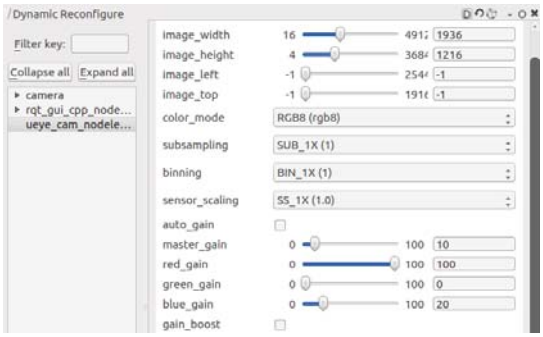

Fig. 1. The graphic user interface of ROS for adjust gains.

The color gains were factory pre-calibrated to a daylight value (5500 Kelvin) [3]. We experimentally adjusted them, including red, green, and blue (in RGB color space), to make the underwater object colors and the actual ones similar. To achieve this, we put a color board with red, green, blue, yellow and black color lines into a swimming pool. Then, we adjusted the color gain until the monitor shows the similar color images between the underwater board and the on-shore one. Moreover, we adjusted the master gain to 10 that means 10 percent of the maximum possible gain. The master gain value depends on the image sensor [3]. The master gain is the analog control that is performed in the sensor and achieves better results than software based image correction [3]. The graphic user interface of ROS [8] for adjust gains is shown in Fig. 1.

III. IMAGE ENHANCEMENT

Image enhancement consists of 2 processes, the first is color image enhancement and the second is grayscale image enhancement. The choice between the processes depends on what type of the final image we want.

A. Color image enhancement

In this section, we apply the contrast limited adaptive histogram equalization (CLAHE) on L of L*A*B to enhance the image quality and smoothing the underwater image [10]. The output is shown in Fig. 2.

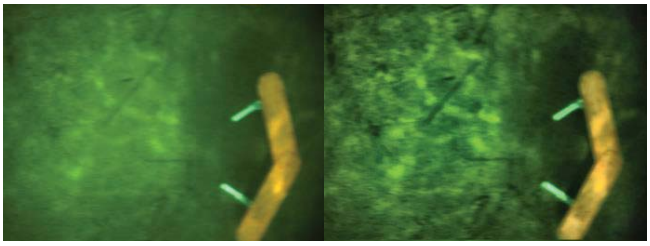

Fig. 2. The input image (left) and the output image (right) passed CLAHE.

B. Grayscale Enhancement

We also apply histogram equalization to the grayscale images for some tasks that requires the detection of the foreground objects. This process increases the intensity of the foreground objects to make them more suitable for further processing. Fig. 3 (left) shows an original image while the Fig. 3 (right) illustrates its enhanced version.

Fig. 3. The input image (left) and the output image (right) passed histogram equaliztion.

IV. GRAYSCALE IMAGE TO BINARY IMAGE

In the Shoot Craps (dice detection) mission in the 2018 International RoboSub Competition [5], we have to detect the number of dot on the dice face. We developed a novel adaptive threshold to convert gray-scale images into their corresponding binary format. The adaptive threshold equation, as in (1).

$$Threshold = \frac{\sum_{i,j=0,0}^{r,c} V_{i,j}}{r\ x\ c}\ x\ Constant \qquad (1)$$

Where Threshold is used to classify the pixel values that should be 0 or 255, r is rows of image, c is columns of image, and V is value of HSV color space.

We did similar experiment as in the grayscale enhancement to figure out the appropriate threshold constant. We have found that the value of 0.5 provides the best result for the experimental dataset.

Moreover, we compared the result of our approach with the other algorithms such as Simple Thresholding (V = 127), Adaptive Mean Thresholding, and Adaptive Gaussian Thresholding. [6] With the input grayscale image shown in Fig. 4, the compared result from each algorithm is illustrated in Fig. 5. Our algorithm gives higher contrast of the dots on each dice face than the others.

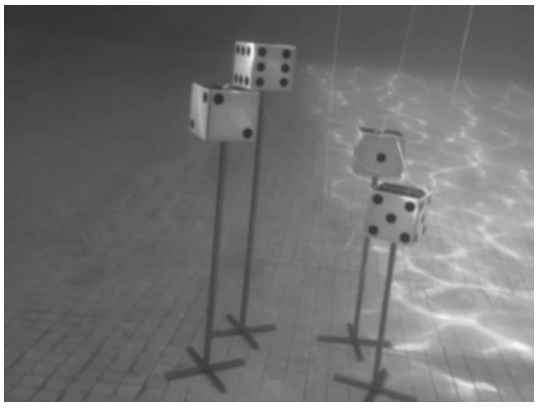

Fig. 4. The input grayscale image.

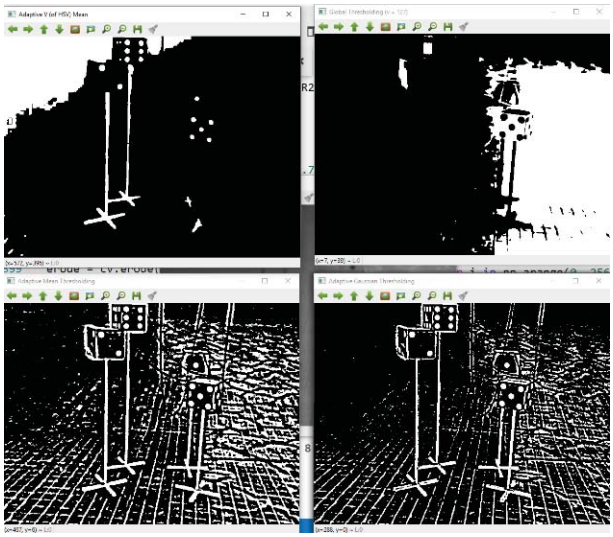

Fig. 5. The result of compared threshold algorithms.

V. COLOR DETECTION

For normal navigation which requires the processing of color images. We converted the images from RGB format to HSV (Hue, Saturation, Value) format because HSV color space is more intuitive to how people experience color than the RGB color space. [7] As the hue (H) varies from 0 to 1.0 (or from 0 to 180 in OpenCV), the color varies from red through yellow, green, cyan, blue, magenta, back, and finally red again. This makes the red color has the values at both 0 and 1.0. As the saturation (S) varies from 0 to 1.0 (or from 0 to 255 in OpenCV), the corresponding color (hue) varies from unsaturated (lighter) to fully saturated (darker). As the value (V) or the brightness varies from 0 to 1.0 (or from 0 to 255 in OpenCV), the corresponding colors become increasingly brighter [7].

After that, the image pixels are classified by a predefined HSV range to create the mask of the relevant object. The HSV range was specified by human tuning through a GUI software, which was developed by the main author using the OpenCV library. An example of color detection for an underwater Path [5] is shown in Fig. 6.

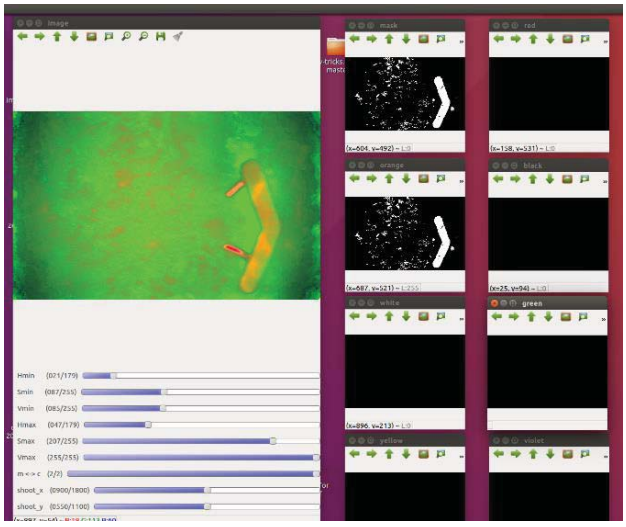

Fig. 6. Graphic user interface of color range.

VI. SHAPE DETECTION

From the section IV, the binary image, called mask [6], is used to detect specific foreground objects in the corresponding picture. We created an object detection algorithm for each task base on mathematics, statistics, probability, and geometry. The algorithm firstly reduces noises in the mask by applying morphological erosion, which results in loss of boundary pixels of an object [9]. Then, we add pixels to the boundary elements with the morphological-dilation algorithm [9] to reconstruct the object in the mask.

Then, we apply a contour, a curve joining all the continuous points having same intensity [4], to mask the detected objects in scene. After that, we implement the Rotated Rectangle [4] to contours. Finally, we define the 4 conditions to classify Path as:

1. The contour areas having more than 2.5 percent of image size.
2. The area ratio between contour and Rotated Rectangle being more than 0.45 and less than 0.6.
3. Having 3 or 4 triangles inside of Rotated Rectangle.
4. The ratio between the longest width and Height ratio being more than 2.5.

Fig. 7 shows the mask consist of Path [5] and noises. After passing mask cleaning process and detection conditions, described above, the object detection algorithm gives the result as shown in Fig. 8. The results have proved that our algorithm is able to distinguish between noise and the object-under-interest very well.

Fig. 7. The mask of Path.

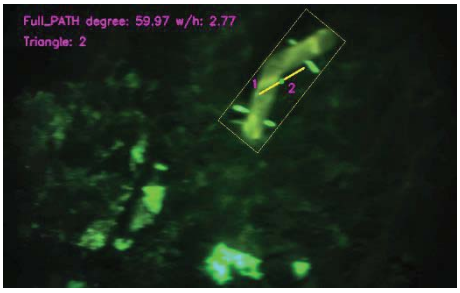

Fig. 8. The result of shape detection to detect Path.

VII. EXPERIMENTAL RESULT

In this section, we explain the total processes of the underwater computer vision of the ZEABUS AUV and the experimental result. From all mission, we select Enter Casino (Gate) and Shoot Craps (Dice) [5] as two examples for our explanation.

In the Enter Casino mission, we use 4 steps as:

Step 1: Applying color image enhancement from section III. The result is shown in Fig. 9.

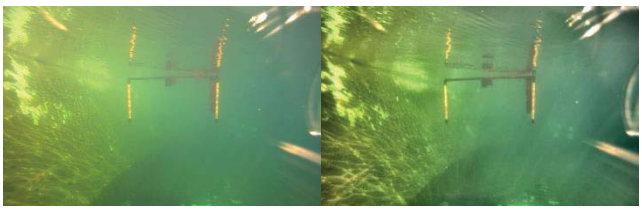

Fig. 9. The input image (left) and the enhanced image (right).

Step 2: Limiting the region and creating mask by the color detection from section V. It selects the colors on both sides of the Gate. The output mask is shown in Fig. 10.

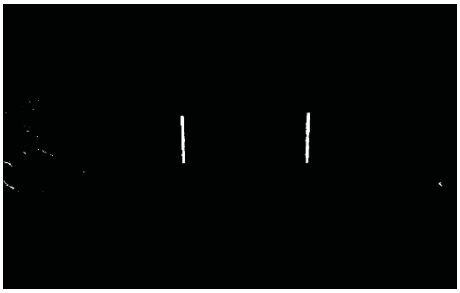

Fig. 10. The mask of Enter Casino (Gate).

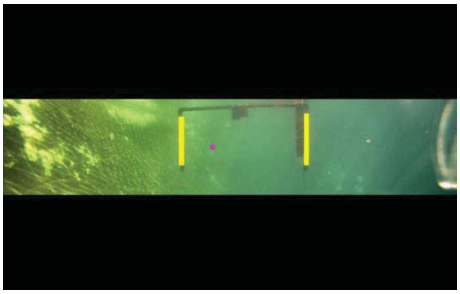

Fig. 11. The result of Enter Casino (Gate).

Step 3: Applying the shape detection with the same condition as in section VI. We also implemented a rule that both poles showing as 2 vertical rectangles should have the center points on vertical axis at the similar levels.

Finally, we calculate the center of the left side of the gate because, in the competition, we must choose the left or right side of the gate before starting the AUV instead of just passing through the gate [5].

In Fig. 11, the gate poles are drawn over by yellow rectangles and the maneuver destination is shown as a red dot.

In another example, the Shoot Craps (Dice) mission, we have 3 steps as:

Step 1: Converting RGB image to grayscale image, then, creating a mask using our adaptive threshold described in section IV. The mask is shown in Fig. 12.

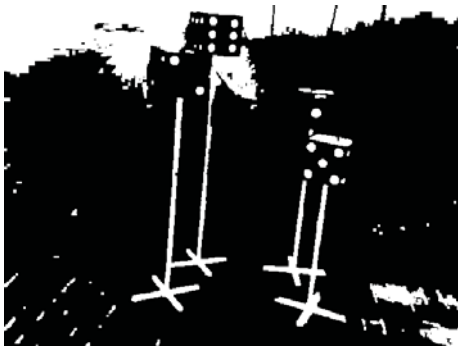

Fig. 12. The mask of Shoot Craps (Dice).

Step 2: Applying the shape detection to classify dots on the dice faces as shown in Fig. 13.

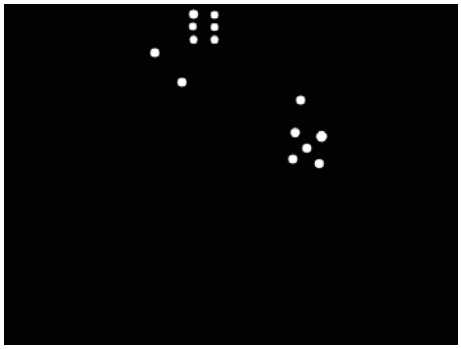

Fig. 13. The mask of shape detection to detect dot on the dice face.

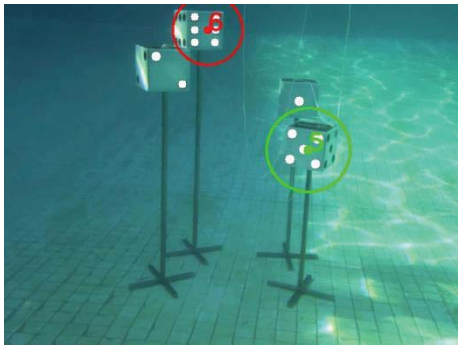

Fig. 14. The result of Shoot Craps (Dice).

Step 3: As stated in the competition rules, we should touch two dice faces that makes the total dots of the faces equal to 11. Therefore, we target the dice faces which have the number of dots as 6 and 5. We performed dot grouping to get the region of interest (ROI) from each dot and check the ROI for a dice by separating the ROI into 9x9 square, then, matched the patterns of dots of the real dice to find the dice face of 5 and 6 dots. The result shown in Fig. 14.

VIII. Conclusion

Underwater computer vision is developed for the 2018 International RoboSub Competition and Singapore AUV Challenge 2018. The algorithms in this paper, are designed to appropriate for each mission and tested in simulation and real environment.

References

[1] D. L. Rizzini, F. Kallasi, F. Oleari, and S. Caselli, “Investigation of vision-based underwater object detection with multiple datasets,” in Int. J. Adv. Robot. Syst., January 1, 2015. Accessed on: Aug. 28, 2018. [Online]. doi: 10.5772/60526

[2] G.L. Foresti, and S. Gentili, “A vision based system for object detection in underwater images,” in Int. J. Pattern Recogn., March 2000. Accessed on: Sep. 2, 2018. [Online]. doi : 10.1142/S021800140000012X

[3] D. Diezemann, and M. Gentele, “Techtip: brighter images thanks to gain,” IDS Imaging Development Systems GmbH, 2015. [Online]. Available: https://en.ids-imaging.com/tl_files/downloads/techtip/TechTip_Gain_EN.pdf. [Accessed: Sep. 4, 2018].

[4] M. Poorani, T. Prathiba, and G. Ravindran, “Integrated feature extraction for image retrieval,” IJCSMC, vol. 2, pp.28 – 35, February 2, 2013. Accessed on: Aug. 30, 2018. [Online]. Available: https://www.ijcsmc.com/docs/papers/February2013/V2I2201307.pdf

[5] “21st Annual international robosub competition,” July 17, 2018. [Online]. Available: https://www.robonation.org/sites/default/files/2018%20RoboSub_2018%20Mission%20and%20Scoring_v01.50.pdf. [Accessed: Aug. 25, 2018].

[6] A. Mordvintsev & K. R. Abid, “OpenCV-Python tutorials documentation,” November 5, 2017. [Online]. Available: https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf. [Accessed: Aug. 20, 2018].

[7] S. Kolkur, D. Kalbande, P. Shimpi, C. Bapat, and J. Jatakia, “Human skin detection using rgb, hsv and ycbcr color models,” ICCASP/ICMMD-2016, Atlantis Press, vol. 137, pp. 324-332, 2017. Accessed on: Sep. 4, 2018. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1708/1708.02694.pdf

[8] S. Cousins, "Welcome to ROS Topics [ROS Topics]," in IEEE Robotics & Automation Magazine, vol. 17, no. 1, pp. 13-14, March 2010. Accessed on: Sep. 12, 2018. [Online]. doi: 10.1109/MRA.2010.935808

[9] S. Ravi, and A. M. Khan, “Morphological operations for image processing: understanding and its applications,” Int. Conf. on VLSI, Commun. and Signal Process., December 2013. Accessed on: Sep. 10, 2018. [Online]. Available: https://www.researchgate.net/publication/272484795_Morphological_Operations_for_Image_Processing_Understanding_and_its_Applications

[10] R. Kaur, and D. Saini, “Image enhancement of underwater digital images by utilizing L*A*B* color space on gradient and CLAHE based smoothing,” in Int. J. of Comput. Appl., March 2015. Accessed on: Sep. 13, 2018. [Online]. doi: 10.5120/cae2016652166